Don't Fear the Bit Flips: Robust Linear Prediction Through Informed Channel Coding

Frederic Sala [1] **Shahroze Kabir** [1] **Lara Dolecek** [1] **Guy Van den Broeck** [1]

Abstract

Traditionally, data storage systems provide error-correction and data integrity techniques in an independent layer, with the goal of protecting all the data equally regardless of the application. In the context of machine learning systems, this strategy is not appropriate: errors in a few features may prove to be critically important (with respect to the algorithm output), while many errors may have little or no effect. This work takes a different direction: we allow ML algorithms to talk to error-correction schemes, with the goal of making algoritms robust to storage noise. We introduce several novel problems, provide an efficient solution to estimate the noise-induced change in the algorithm output for linear models, and show how to optimize error-correction codes to minimize error effects for fixed overhead.

1. Introduction

The standard approach to handling noise and faults in data storage systems is to provide a separate abstraction layer where error and fault correction techniques are applied to maintain the integrity of the data. The goal of this strategy is to ensure that all of the data is reliable; the correction layer is agnostic to the *use* of the data.

This solution may not be suitable for big data systems. Uniformly protecting the data is equivalent to minimizing the error rate measured at the *input* to some algorithm; this rate does not directly translate to the effect on the *output*. Even a very large number of errors may not have any impact on, for example, a classifier output. On the other hand, an error in a single feature (out of many) may have a dramatic impact, even though the input error rate remains very low.

[1]University of California Los Angeles, Los Angeles, California, USA. Correspondence to: Frederic Sala <fredsala@ucla.edu>.

Proceedings of the ICML 2017 Workshop on Reliable Machine Learning in the Wild, Sydney, Australia, 2017. Copyright 2017 by the author(s).

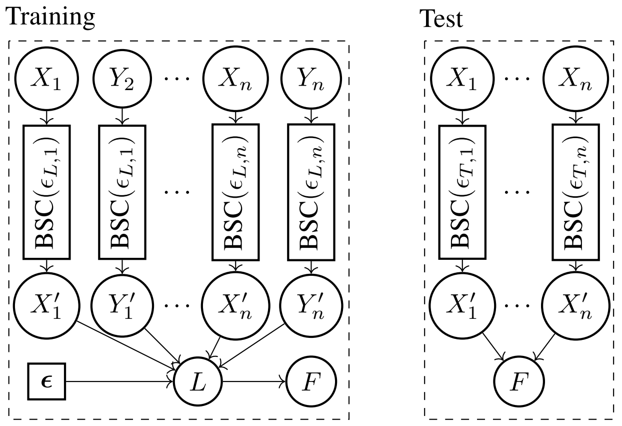
Training
Test
X_1
Y_2
X_n
Y_n
$\mathrm{BSC}(\epsilon_{L,1})$
$\mathrm{BSC}(\epsilon_{L,n})$
$\mathrm{BSC}(\epsilon_{T,1})$
$\mathrm{BSC}(\epsilon_{T,n})$
X'_1
Y'_1
X'_n
Y'_n
ϵ
L
F

Figure 1. Noisy system. Training phase (left) uses binary features and labels (X, Y) passed through a noisy binary symmetric channel BSC(ϵ): X' is equal to X with probability $1 - \epsilon$ and equal to $\bar{X}$ with probability ϵ. Algorithm L can use noise statistics $\boldsymbol{\epsilon}$ to produce a model F minimizing the impact of noise. Test phase (right) also operates on noisy features X' without observing the true values X.

This observation inspires us to consider a cross-layer approach for error-protection in machine learning systems, with the goal of reducing the effect of noise on the algorithm output. This novel problem bridges machine learning and the traditional fields concerned with the reliability of data, such as coding and information theory.

A general setting is shown in Figure 1. Noise can be found in the training data and labels (X, Y) (left). The learned model F is affected by this noise, but can also be modified to be noise-aware by using the noise statistics $\boldsymbol{\epsilon}$. This model operates on noisy test data (right). The setting leads to the following hierarchy of problem settings, in increasing order of intractability:

1. Fixed model (learned from noiseless training data, i.e., $\epsilon_{L,i} = 0$) operating on noisy test data,
2. Noise-aware model (learned from noiseless training data and noise statistics $\boldsymbol{\epsilon}$), produced to minimize expected noise loss on the output,
3. Noise-aware model learned from noisy training data.

Table 1. Noise loss (probability of incorrect classification) for various redundancy allocations.

Parameters				Noise loss by protection strategy			
$p_0(x_1)$	$p_0(x_2)$	$p_1(x_1)$	$p_1(x_2)$	Uniform	None	All x_1	All x_2
0.10	0.11	0.90	0.89	0.037	0.10	**0.009**	0.10
0.20	0.19	0.70	0.70	**0.036**	0.095	0.054	0.054

We largely concentrate on the first setting (which is quite challenging) while briefly discussing the other problems.

Given one of the problem settings, the first task is to evaluate the impact of the noise by measuring the loss (expected noise loss, or ENL) on the output compared to a noiseless version with the same features. For problems 2 and 3, we can modify L to produce a model minimizing this loss. However, in all cases, we can *modify the ϵ's* through application of error-correcting codes. This is the key insight that enables us to bring together machine learning and coding theory. Naturally, forcing $\epsilon \to 0$ minimizes the loss; however, this may require an unacceptable amount of coding overhead. Instead, we focus on optimizing the code (with respect to the loss function) for a fixed overhead. We illustrate the main ideas with the following toy example:

2. Motivating Example

Consider a naive Bayes classifier with $n = 2$ features X_1, X_2, noisy versions X_1', X_2', and uniform prior on the class. The noise parameter is $\epsilon = 0.1$ by default. Let $p_i(x_j) := p(x_j = 0|C = i)$. We allocate 4 additional redundancy bits for protection; uniform protection gives 2 such bits to each feature, while protection on X_i alone grants all 4 bits to X_i (the result of protection is a reduction of the ϵ_i noise parameter for feature i.)

In the first row of Table 1, X_1 contains more information about class C compared to X_2. Observe that allocating all bits to X_1 yields a lower loss than equal protection or protection on X_2. Conversely, in the last row, even though X_2 contains more information about C, the loss is minimized by a uniform allocation of redundancy bits. The average loss between noisy and noiseless classifier output (equivalent to the probability of changing the classification) varies for different values of the $p_i(x_j)$'s and noise parameters. These observations imply that we cannot directly examine the conditional probabilities to decide how to allocate redundancy - we need an informed coding allocation strategy.

3. Related Work

There is a vast literature on robust machine learning algorithms. For example, Ramoni & Sebastiani (2001) considers missing training set data (missing features or labels) and is concerned with the impact on classifier accuracy. In Provost & Fawcett (2001), varying operative conditions and their effect on classifiers are studied. An experimental study of algorithms with synthetic noise corrupted datasets is performed by Kalapanidas et al. (2003). Deleted features are also tackled by Dekel & Shamir (2008) and Globerson & Roweis (2006); the latter proposes a game-theoretic approach to avoid over-reliance on a deleted feature.

Our approach (more detail in Mazooji et al. (2016) and Sala et al. (2017)) fundamentally differs from such works. In our problem, the noise occurs after feature generation (e.g., because the features are stored on noisy data storage devices). However, the system has the ability to protect the algorithm output with tailored error-correction strategies.

At the same time, our work stands in contrast to existing research on channel coding for data protection. The goal in these works is to preserve the data being stored without considering the application. That is, error-correction forms a separate abstraction layer. Works in this area propose error-correction for disk drives (Riggle & McCarthy, 1998), write-once memories (Rivest & Shamir, 1982), RAID architectures (Blaum et al., 1995), non-volatile memories such as flash (Cassuto et al., 2010; Dolecek & Sala, 2016), solid-state drives (Zhao et al., 2013), and distributed storage (Weatherspoon & Kubiatowicz, 2002), (Dimakis et al., 2010).

4. Noise in Linear Models

We consider algorithms resulting in linear models in problem setting 1. For simplicity, we use binary features, although our results easily extend to non-binary features. Let $[n] := \{1, 2, \ldots, n\}$. Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be a test point. Our running examples will be linear regression and linear classifiers; the classifier will be in the form of naive Bayes, though the results equivalently apply to logistic regression, other hyperplane-based classifiers, etc.

Noise Model. We employ a simple noise model: we define a *noise parameter vector* $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2, \ldots, \epsilon_n)$ with $0 \leq \epsilon_i \leq 1/2$ for $1 \leq i \leq n$. The binary feature X_i is flipped to its opposite value $\overline{X_i}$ with probability ϵ_i and stays unchanged with probability $1 - \epsilon_i$. We express the resulting *error vector* as $\mathbf{E} = (E_1, E_2, \ldots, E_n)$, where $E_i = 1$ if an error has occurred for feature i and 0 otherwise. If errors occur at positions in $S \subseteq [n]$ in $\mathbf{E}$, then the probability of $\mathbf{E}$ is given by $Pr(\mathbf{E}) = \prod_{i \in S} \epsilon_i \prod_{i \notin S}(1 - \epsilon_i)$. We denote the algorithm operating on the noisy features as $F(\mathbf{x}, \mathbf{E}) = F(\mathbf{x} \oplus \mathbf{E})$.

Noise Loss Measures. We examine the effect of the feature bit errors on the output of the algorithm. Let $\ell(F(\mathbf{e}), F(\mathbf{x}, \mathbf{e}))$ be the noise loss between the output of the noiseless and noisy versions of the algorithms. We are interested in evaluating the *expected noise loss* (ENL)

$\mathbb{E}_{\mathbf{E}}[\ell(F(\mathbf{X}), F(\mathbf{X}, \mathbf{E}))]$.

Conditioning on $\mathbf{E}$ and letting $S_{\mathbf{e}}$ be the positions i where $e_i = 1$, the ENL can be written as

$$\sum_{S_{\mathbf{e}} \subseteq [n]} \ell(F(\mathbf{X}), F(\mathbf{X}, \mathbf{e})) \prod_{i \in S_{\mathbf{e}}} \epsilon_i \prod_{i \notin S_{\mathbf{e}}} (1 - \epsilon_i). \quad (1)$$

We are interested in the empirical ENL $\mathbb{E}_{\mathbf{E}}[\ell(F(\mathbf{x}), F(\mathbf{x}, \mathbf{E}))]$. A natural choice for ℓ is the L_1 norm: $\ell(F(\mathbf{x}), F(\mathbf{x}, \mathbf{E})) = |F(\mathbf{x}) - F(\mathbf{x}, \mathbf{E})|$. For classification problems, this 0/1 loss reduces to the *classification change probability*, that is, the the probability that the noiseless point $\mathbf{X}$ and the noisy version $\mathbf{X}'$ have differing classification. For linear regression problems, the L_2 norm is also of interest.

For linear models, the ENL can be written, for a function g determined by the algorithm, as the discrete integral

$$\mathbb{E}_{\mathbf{E}}[\ell(F(\mathbf{x}), F(\mathbf{x}, \mathbf{E}))] = \sum_{S \in [n]} g\left(\sum_{i \in S} D_i\right) \prod_{i \in S} \epsilon_i \prod_{i \notin S} (1 - \epsilon_i), \quad (2)$$

for certain choices of D_i. In linear regression, $|F(\mathbf{x}) - F(\mathbf{x}, \mathbf{E})| = |\mathbf{a}^T\mathbf{x} - \mathbf{a}^T(\mathbf{x} \oplus \mathbf{E})|$, so that $D_i = a_i(-1)^{x_i}$ for $1 \le i \le n$. Thus for linear regression, $g(v) = |v|$. For Naive Bayes, set the loss term D_j to be the difference

$$D_j = B_j - A_j = \log((1 - \alpha_j)/(1 - \beta_j)) - \log(\alpha_j/\beta_j),$$

where $\alpha_i = p(x_i|C = 0)$ and $\beta_i = p(x_i|C = 1)$. Then, it can be shown that $g(v) = \mathbb{1}\{v < T\}$, i.e., for linear classifiers

$$\mathbb{E}_{\mathbf{E}}[\ell(F(\mathbf{x}), F(\mathbf{x}, \mathbf{E}))] = \sum_{S \in [n]} \mathbb{1}\left\{\sum_{i \in S} D_i < T\right\} \prod_{i \in S} \epsilon_i \prod_{i \notin S} (1 - \epsilon_i). \quad (3)$$

5. Computing the Expected Noise Loss

Computing the ENL is a significant challenge. As a warm-up, we compute the ENL under the L_2 norm for linear regression. It is not hard to show that

$$\mathbb{E}_{\mathbf{E}}[||\mathbf{a}^\intercal\mathbf{x} - \mathbf{a}^\intercal\mathbf{x}'||] = \mathrm{Tr}(\mathbf{a}\mathbf{a}^\intercal Cov(\mathbf{x} - \mathbf{x}')) + \mathbb{E}_{\mathbf{E}}[\mathbf{x} - \mathbf{x}']^\intercal \mathbf{a}\mathbf{a}^\intercal \mathbb{E}_{\mathbf{E}}[\mathbf{x} - \mathbf{x}'].$$

It is easy to compute $\mathbb{E}_{\mathbf{E}}[\mathbf{x} - \mathbf{x}']$. If $x_i = 0$, the expectation is $-\epsilon_i$, while if $x_1 = 1$, it is ϵ_i, so that $\mathbb{E}_{\mathbf{E}}[\mathbf{x} - \mathbf{x}'] = [(-1)^{x_1+1}\epsilon_1, \ldots, (-1)^{x_n+1}\epsilon_n]^\intercal$. Moreover, a similar solution can be found for Problem 2 for linear regression: it is possible to find the optimal coefficients $\mathbf{a}$ that minimize the expected L_2 loss as a function of the ϵ's.

Algorithm 1 Expected Noise Loss Approximation

```
  Input: Loss terms $D_1, D_2, \ldots, D_n$, Error probabilities $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$, Target $T$ (in classification problems), Number of buckets in quantization scheme $k$, Model function $g$
  Output: Expected noise loss approximation ENL$_{app}$
  Initialize $S_1, \ldots, S_k$ to 0, ENL$_{app}$ to 0
  for $j = 1$ to $n$ do
    if $D_j \in$ [Interval$_i^{\text{begin}}$, Interval$_i^{\text{end}}$) then
      $N_i \leftarrow N_i + 1$, $\epsilon_{i,N_i} \leftarrow \epsilon_j$
    end if
  end for
  expand $G[i,j] = \prod_{i=1}^{k}\prod_{j=1}^{N_i}((1 - \epsilon_{i,j}) + \epsilon_{i,j}yz^i)$
  for $i = 0$ to $n$ do
    for $j \ge 0$ do
      ENL$_{app}$ $\leftarrow$ ENL$_{app}$ + $g_{i,j}(G[i,j])$
    end for
  end for
```

Algorithm 2 Channel Code Optimization

```
  Input: Test points $\mathbf{x}_1, \mathbf{x}_2 \ldots, \mathbf{x}_t$, Noise vector $\boldsymbol{\epsilon} = (\epsilon, \ldots, \epsilon)$, redundancy budget $r$
  Output: Optimized redundancy allocation vector $\mathbf{r}^r$
  Initialize $\mathbf{r}(0)$ to 0
  for $j = 0$ to $r - 1$ do
    $i \leftarrow \arg\min_i \frac{1}{t}\sum_{k=1}^{t} \mathrm{ENL}_{app}(\mathbf{x}_k, \mathbf{E}^{\mathbf{r}(j)+\mathbf{1}(i)})$)
    $\mathbf{r}(j+1) \leftarrow \mathbf{r}(j) + \mathbf{1}(i)$
  end for
```

In contrast to the L_2 loss, which offers the convenient factorization in above, the general problem and the L_1 loss function version are difficult to compute:

Proposition 1. *Expression (2) is #P-hard to compute, even for an efficiently computable function g.*

Proposition 2. *Expression (3) is NP-hard to compute.*

To deal with this, we provide a lightweight, efficient approximation of the ENL relying on a discretization scheme. The key idea is to quantize the loss terms D_i into k intervals, for k a constant. By a clever choice of quantization, we induce a structure that enables us to approximate the ENL in no more than $O(n^2 \log n)$ operations. The approximation is described in Algorithm 1. For linear regression, set $g_{\ell,m}(v) = |\frac{m}{k-1}D_I + \ell(D_{min} - \frac{D_I}{k-1})| \times v$. For linear classifiers, set $g_{\ell,m}(v) = \mathbb{1}\{m < T'\} \times v$ where $T'(\ell) := \frac{k-1}{D_I}\left(T - \ell\left(D_{min} - \frac{D_I}{k-1}\right)\right)$. Details on how to perform the quantization are found in the longer version of this work (Sala et al., 2017). The key result is that the approximation can be performed efficiently:

Theorem 1. *The ENL approximation in Algorithm 1 requires no more than $O(k^2n^2\log(kn))$ computations of g.*

Table 2. Expected Noise Loss (ENL) for classification and regression

	Movie dataset, feature set 1			Movie dataset, feature set 2			Voting dataset			Bike rent dataset		
R	Uniform	Optimal	Ratio	Uniform	Optimal	Ratio	Uniform	Optimal	Ratio	Uniform	Optimal	Ratio
1	0.1263	0.1025	1.233	0.1011	0.0866	1.167	0.0564	0.0414	1.363	504.7	396.5	1.273
2	0.0442	0.0303	1.459	0.0348	0.0302	1.152	0.0318	0.0238	1.339	131.6	102.3	1.286
3	0.0147	0.0088	1.671	0.0115	0.0100	1.151	0.0190	0.0143	1.332	35.28	27.32	1.291
4	0.0047	0.0026	1.903	0.0038	0.0033	1.154	0.0115	0.009	1.356	10.16	7.982	1.273
5	0.0016	0.0008	2.101	0.0012	0.0011	1.158	0.0070	0.0050	1.391	3.486	2.884	1.209
6	0.0005	0.0002	2.255	0.0004	0.0003	1.160	0.0042	0.0029	1.433	1.682	1.515	1.110
7	0.0001	7.7E-05	2.398	0.0002	0.0001	1.162	0.0026	0.0018	1.474	1.189	1.142	1.041
8	6.3E-05	2.5E-05	2.539	4.9E-05	4.2E-05	1.164	0.002	0.0011	1.518	1.053	1.040	1.013
9	2.2E-05	8.1E-06	2.672	1.7E-05	1.5E-05	1.165	0.001	0.0006	1.563	1.015	1.011	1.004
10	7.4E-06	2.6E-06	2.811	5.8E-06	5.0E-06	1.165	0.0006	0.0004	1.611	1.004	1.003	1.001

6. Optimized Channel Coding

Now that we can compute the ENL, the next step is is to tailor an error-correction strategy to the characteristics of the features in order to minimize the ENL (reducing the harmful impact of the noise on the output of the algorithm).

Repetition codes. We rely on repetition coding. If the jth feature x_j, corrupted by a bit flip with probability ϵ, is repeated $2r+1$ times for some $r \geq 0$, the probability $\epsilon^{(r)}$ that the feature is decoded incorrectly is equal to the probability that the majority of votes are corrupted: $\epsilon^{(r)} = \sum_{i=r+1}^{2r+1} \binom{2r+1}{i} \epsilon^i (1-\epsilon)^{2r+1-i}$. Repetitions codes are useful due to the flexibility of per-bit independent encoding and decoding. Unlike in classical coding, we have found that such codes can be optimal in our setting.

Coding optimization. We use a total budget of $2r$ redundancy bits. Feature i is represented by $1+2r_i$ copies, and has error probability $\epsilon^{(r_i)}$, for $1 \leq i \leq n$. The noise vector for all n features is $\boldsymbol{\epsilon}^{(\mathbf{r})} = (\epsilon^{(r_1)}, \epsilon^{(r_2)}, \ldots, \epsilon^{(r_n)})$. The corresponding error vector is written $\mathbf{E}^{(\mathbf{r})}$. The values $\mathbf{r} = (r_1, r_2, \ldots, r_n)$ are constrained by $r_1 + r_2 + \ldots + r_n = r$. The goal is to minimize the ENL over t test points $\mathbf{x}_1, \ldots, \mathbf{x}_t$ with respect to $\mathbf{r}$ and the resulting $\boldsymbol{\epsilon}$ vector. We have the following optimization

$$\arg\min_{\mathbf{r}} \frac{1}{t} \sum_{j=1}^{t} \text{ENL}(\mathbf{x}_j, \mathbf{E}^{(\mathbf{r})}) \text{ s.t.} \sum_{i=1}^{n} r_i = r, r_i \geq 0. \quad (4)$$

Consider the following greedy allocation algorithm, using the ENL approximation in Algorithm 1 and performed one redundancy unit (two repeated bits) at a time. After the jth step, we write the redundancy vector as $\mathbf{r}(j) = (r_1(j), r_2(j), \ldots, r_n(j))$. In the $(j+1)$st step, one of the $r_i(j)$ terms is selected and increased by 1. We write $\mathbf{1}(i)$ for the vector $(0, 0, \ldots, 0, 1, 0, \ldots, 0)$ with a 1 in the ith position and 0's elsewhere. Then, the $(j+1)$st step is given by $\arg\min_i \frac{1}{t} \sum_{k=1}^{t} \text{ENL}_{app}(\mathbf{x}_k, \mathbf{E}^{\mathbf{r}(j)+\mathbf{1}(i)}))$ s.t. $1 \leq i \leq n$. The complexity is reduced to that of performing nt ENL computations per each of the r steps. The procedure is given in Algorithm 2.

For linear classifiers, we found that the redundancy allocation function is submodular, and thus the greedy algorithm is close to optimal. This requires two properties: (i) monotonicity (which holds with high probability when the test and training distributions are the same) and (ii) the 'diminishing returns' property, which follows from monotonicity.

Experiments. We demonstrate the benefits of our scheme on several datasets: binary classification uses a voting dataset (Lichman, 2013) and a movie review dataset (Bekker et al., 2015), and linear regression uses a bike rental dataset (Gam, 2013). In all cases, we used $k = 50$ for the number of buckets in our ENL approximations. For the movie reviews set we used two sets of $n = 20$ features and 250 test data points. The ϵ parameter was 0.2 for the first 2 datasets (a very high amount of noise that can nevertheless be handled through channel coding). Table 2 shows the ENL for uniform and optimized assignment of redundancy bits given a budget of $2nR$ bits. For the classification setting, the values can be interpreted as the probability of switching the classification due to noisy test features; for regression, the values can be thought of as a type of distortion on the output. We also report the ratio of the ENL between the uniform and optimized coding strategies; this ratio represents the benefits of our coding strategy. We observe performance improvements of between 10% and 300% for various data sets.

7. Conclusion

We proposed a new class of problems where data protection is tailored to take advantage of the application. For the problem setting with linear models operating on noisy test data, we introduced an efficient scheme to evaluate the noise loss and an optimized coding scheme targeted at minimizing the noise loss based on submodular optimization. Our informed approach for redundancy allocation is among the first principled methods combining coding theory with machine learning. Several interesting problems remain for future work: non-linear models, other loss functions, and the general cases of Problems 2 and 3.

Acknowledgements

This work is partially supported by the NSF (#IIS-1657613, #IIS-1633857, CAREER CCF-1150212, CCF-1527130), and DARPA (#N66001-17-2-4032).

References

Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2013. ISSN 2192-6352. doi: 10.1007/s13748-013-0040-3.

Bekker, J., Davis, J., Choi, A., Darwiche, A., and Van den Broeck, G. Tractable learning for complex probability queries. In *Proceedings of Conference on Neural Information Processing Systems (NIPS)*, Montreal, Canada, 2015.

Blaum, M., Brady, J., Bruck, J., and Menon, J. EVEN-ODD: an efficient scheme for tolerating double disk failures in RAID architectures. *IEEE Transactions on Computers*, 44(2):192–202, 1995.

Cassuto, Y., Schwartz, M., Bohossian, V., and Bruck, J. Codes for asymmetric limited-magnitude errors with application to multilevel flash memories. *IEEE Transactions on Information Theory*, 56(4):1582–1595, 2010.

Dekel, O. and Shamir, O. Learning to classify with missing and corrupted features. In *Proc. 25th International Conference on Machine Learning (ICML 2008)*, Helsinki, Finland, 2008.

Dimakis, A. G., Godfrey, P. B., Wu, Y., Wainwright, M. J., and Ramchandran, K. Network coding for distributed storage systems. *IEEE Transactions on Information Theory*, 56(9):4539–4551, 2010.

Dolecek, L. and Sala, F. Channel coding methods for non-volatile memories. *Foundations and Trends in Communications and Information Theory*, 13(1):1–136, 2016.

Globerson, A. and Roweis, S. Nightmare at test time: robust learning by feature deletion. In *Proc. 25th International Conference on Machine Learning (ICML 2006)*, Pittsburgh, PA, 2006.

Kalapanidas, E., Avouris, N., Craciun, M., and Neagu, D. Machine learning algorithms: a study on noise sensitivity. In *Proc. 1st Balcan Conference in Informatics*, pp. 356–365, Thessaloniki, Greece, 2003.

Lichman, M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Mazooji, K., Sala, F., Van den Broeck, G., and Dolecek, L. Robust channel coding strategies for machine learning data. In *Proc. Allerton Conf. Comm., Control, and Computing*, pp. 609–616, 2016.

Provost, F. and Fawcett, T. Robust classification for imprecise environments. *Journal of Machine Learning Research*, 42(3):203–231, 2001.

Ramoni, M. and Sebastiani, P. Robust bayes classifiers. *Artificial Intelligence*, 125(1-2):209–226, 2001.

Riggle, C. M. and McCarthy, S. G. Design of error correction systems for disk drives. *IEEE Transactions on Magnetics*, 34(4):2362–2371, 1998.

Rivest, R. L. and Shamir, A. How to reuse a “write-once memory”. *Information and Control*, 55(1-3):1–19, 1982.

Sala, F., Kabir, S., Van den Broeck, G., and Dolecek, L. Don’t fear the bit flips: Optimized coding strategies for binary classification. 2017. URL https://arxiv.org/abs/1703.02641.

Weatherspoon, H. and Kubiatowicz, J. Erasure coding vs. replication: A quantitative comparison. In *Proceedings of First International Workshop on Peer-to-Peer Systems (IPTPS ’01)*, 2002.

Zhao, K., Zhao, W., Sun, H., Zhang, T., Zhang, X., and Zheng, N. LDPC-in-SSD: Making advanced error correction codes work effectively in solid state drives. In *Proceedings of Conference on File and Storage Technologies (FAST ‘13)*, San Jose, CA, 2013.